TUGAS TEORI DOKUMENTASI

UAS KRIPTOGRAFI

NAMA : OKTAVANUS

NIM : 201581104

KELOMPOK : 02

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS ESA UNGGUL 2017

1. DES
   a. Fungsi DES

      Algoritma DES (Data Encryption Standard) adalah algoritma enkripsi yang paling banyak digunakan di dunia. Selama bertahun-tahun, dan di antara banyak orang, "pembuatan kode rahasia" dan DES telah serupa. Dan terlepas dari kudeta baru-baru ini oleh Electronic Frontier Foundation dalam menciptakan mesin $ 220.000 untuk memecahkan pesan terenkripsi DES, DES akan tinggal di pemerintahan dan perbankan selama bertahun-tahun yang akan datang melalui versi extended-extended. Sehingga, secara umum fungsi algoritma ini adalah diciptakan untuk merahasiakan suatu pesan agar tidak bisa dibaca oleh pihak yang tidak absah. berpedoman dari fungsi umum tersebut, sehinga pesan yang dikirimkan aman dan tidak dapat dibajak.
   b. Cara Kerja DES

      DES adalah blok cipher - artinya beroperasi pada blok plaintext dengan ukuran tertentu (64-bit) dan mengembalikan blok ciphertext dengan ukuran yang sama. Jadi, DES menghasilkan permutasi di antara 2 ^ 64 (baca ini sebagai: "2 sampai kekuatan ke-64") pengaturan yang mungkin dari 64 bit, yang masing-masing dapat berupa 0 atau 1. Setiap blok 64 bit dibagi menjadi dua blok masing masing 32 bit, setengah blok kiri L dan kanan setengah R. (Pembagian ini hanya digunakan dalam operasi tertentu.)

      Contoh: Misalkan M adalah pesan teks biasa M = 0123456789ABCDEF, di mana M berada dalam format heksadesimal (basis 16). Menulis ulang M dalam format biner, kita mendapatkan blok teks 64-bit:

      M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111
      L = 0000 0001 0010 0011 0100 0101 0110 0111
      R = 1000 1001 1010 1011 1100 1101 1110 1111

      Bit pertama dari M adalah "0". Bit terakhir adalah "1". Kita membaca dari kiri ke kanan.

      DES beroperasi pada blok 64-bit dengan menggunakan ukuran kunci 56-bit. Kunci sebenarnya tersimpan 64 bit, tapi setiap bit ke 8 pada kunci tidak digunakan (yaitu bit bernomor 8, 16, 24, 32, 40, 48, 56, dan 64). Namun, kita tetap akan menghitung bit dari 1 sampai 64, ke kiri ke kanan, dalam perhitungan berikut. Tapi, seperti yang akan Anda lihat, delapan bit yang baru disebutkan bisa dieliminasi saat kita membuat subkunci.

      Contoh: Misalkan K adalah kunci heksadesimal K = 133457799BBCDFF1. Ini memberi kita kunci biner (setting 1 = 0001, 3 = 0011, dll, dan mengelompokkan bersama setiap delapan bit, yang terakhir tidak ada dalam kelompok masing-masing):

      K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

      Algoritma DES menggunakan langkah-langkah berikut:

      Langkah 1: Buat 16 subkunci, masing-masing panjang 48-bit.

      Kunci 64-bit dimodifikasi menurut tabel berikut, PC-1. Karena entri pertama dalam tabel adalah "57", ini berarti bahwa bit ke-57 dari kunci asli K menjadi bit pertama dari kunci permutasi K +. Bit ke 49 dari kunci asli menjadi bit kedua dari kunci yang dimodifikasi. Bit ke 4 dari kunci asli adalah bit terakhir dari kunci yang dimodifikasi. Perhatikan hanya 56 bit dari tombol asli yang muncul pada tombol permutasi.

PC-1

57	49	41	33	25	17	9
1	58	50	42	34	26	18
10	2	59	51	43	35	27
19	11	3	60	52	44	36
63	55	47	39	31	23	15
7	62	54	46	38	30	22
14	6	61	53	45	37	29
21	13	5	28	20	12	4

Contoh: Dari tombol 64-bit asli

K = 00010011 00110100 01010111 01111001 10011011 10111100 11011111 11110001

kita mendapatkan permutasi 56-bit

K + = 1111000 0110011 0010101 0101111 0101010 1011001 1001111 0001111

Selanjutnya, pisahkan kunci ini ke bagian kiri dan kanan, C0 dan D0, di mana masing-masing bagian memiliki 28 bit.

Contoh: Dari kunci permutasi K +, kita dapatkan

C0 = 1111000 0110011 0010101 0101111

D0 = 0101010 1011001 1001111 0001111

Dengan C0 dan D0 didefinisikan, sekarang kita membuat 16 blok Cn dan Dn, 1 <= n <= 16. Setiap pasangan blok Cn dan Dn dibentuk dari pasangan sebelumnya Cn-1 dan Dn-1, masing-masing, untuk n = 1, 2, ..., 16, dengan menggunakan jadwal "shift kiri" berikut dari blok sebelumnya. Untuk melakukan pergeseran kiri, gerakkan masing-masing satu tempat ke kiri, kecuali untuk bit pertama, yang diayunkan ke ujung blok.

Iteration Number	Number of Left Shifts
1	1
2	1
3	2
4	2
5	2
6	2
7	2
8	2
9	1
10	2
11	2
12	2
13	2
14	2
15	2
16	1

Ini berarti, misalnya, C3 dan D3 diperoleh dari C2 dan D2, masing-masing, dengan dua shift kiri, dan C16 dan D16 diperoleh dari C15 dan D15, masing-masing, dengan satu shift kiri. Dalam semua kasus, dengan satu pergeseran kiri berarti rotasi bit satu tempat ke kiri, sehingga setelah satu pergeseran kiri, bit pada posisi 28 adalah bit yang sebelumnya berada pada posisi 2, 3, ..., 28, 1.

Contoh: Dari pasangan pasangan asli C0 dan D0 kita dapatkan:

C0 = 11110000110011001010101011111

D0 = 01010101011001100111100011111

C1 = 11100001100110010101010111111

D1 = 10101010110011001111100011110

C2 = 11000011001100101010101111111

D2 = 01010101100110011110001111101

C3 = 00001100110010101010111111111

D3 = 01010110011001111100011110101

C4 = 00110011001010101011111111100

D4 = 01011001100111100011110101010

C5 = 11001100101010101111111110000

D5 = 011001100111100011110101010 1

C6 = 001100101010111111111000011

D6 = 100110011110001111010101010 1

C7 = 110010101010111111110000110 0

D7 = 011001111000111101010101011 0

C8 = 001010101011111111100001100 11

D8 = 100111100011110101010101100 1

C9 = 010101010111111111000011001 10

D9 = 001111000111101010101011001 1

C10 = 010101011111111100001100110 01

D10 = 111100011110101010101100110 0

C11 = 010101111110000110011001 01

D11 = 110001111010101010110011001 1

C12 = 010111111110000110011001 0101

D12 = 000111101010101011001100 1111

C13 = 011111111000011001100101010 1

D13 = 011110101010101100110011110 0

C14 = 111111100001100110010101010 1

D14 = 111010101010110011001111000 1

C15 = 111110000110011001010101011 1

D15 = 101010101011001100111100011 1

C16 = 111100001100110010101010111 1

D16 = 010101010110011001111000111 1

Kita sekarang membentuk kunci Kn, untuk 1 <= n <= 16, dengan menerapkan tabel permutasi berikut ke masing-masing pasangan CnDn yang digabungkan. Setiap pasangan memiliki 56 bit, namun PC-2 hanya menggunakan 48 di antaranya.

PC-2

14 17 11 24 1 5

3 28 15 6 21 10

23 19 12 4 26 8

16 7 27 20 13 2

41 52 31 37 47 55

30 40 51 45 33 48

44 49 39 56 34 53

46 42 50 36 29 32

Oleh karena itu, bit pertama Kn adalah bit ke 14 dari CnDn, bit kedua ke-17, dan seterusnya, berakhir dengan bit ke-48 Kn menjadi bit 32 dari CnDn.

Contoh: Untuk kunci pertama kita memiliki C1D1 = 1110000 1100110 0101010 1011111 1010101 0110011 0011110 0011110

yang, setelah kita menerapkan permutasi PC-2, jadilah

K1 = 000110 110000 001011 101111 111111 000111 000001 110010

Untuk kunci lain yang kita miliki

K2 =

K3 = 010101 011111 110010 001010 010000 101100 111110 011001

K4 =

K5 = 011111 001110 110000 000111 111010 110101 001110 101000

K6 = 011000 111010 010100 111110 010100 000111 101100 101111

K7 = 111011 001000 010010 110111 111101 100001 100010 111100

K8 = 111101 111000 101000 111010 110000 010011 101111 111011

K9 = 111000 001101 101111 101011 111011 011110 011110 000001

K10 = 101100 011111 001101 000111 101110 100100 011001 001111

K11 = 001000 010101 111111 010011 110111 101101 001110 000110

K12 = 011101 010111 000111 110101 100101 000110 011111 101001

K13 = 100101 111100 010111 010001 111110 101011 101001 000001

K14 = 010111 110100 001110 110111 111100 101110 011100 111010

K15 = 101111 111001 000110 001101 001111 010011 111100 001010

K16 = 110010 110011 110110 001011 000011 100001 011111 110101

Begitu banyak untuk subkunci. Sekarang kita melihat pesan itu sendiri.

Langkah 2: Menyandikan setiap blok data 64-bit.

Ada IP permutasi awal dari 64 bit data pesan M. Ini menata ulang bit-bit sesuai tabel berikut, di mana entri dalam tabel menunjukkan susunan bit baru dari urutan awal mereka. Bit ke-58 M menjadi bit pertama IP. Bit ke-50 M menjadi bit kedua IP. Bit ke 7 dari M adalah bit terakhir dari IP.

IP

58 50 42 34 26 18 10 2

60 52 44 36 28 20 12 4

62 54 46 38 30 22 14 6

64 56 48 40 32 24 16 8

57 49 41 33 25 17 9 1

59 51 43 35 27 19 11 3

61 53 45 37 29 21 13 5

63 55 47 39 31 23 15 7

Contoh: Menerapkan permutasi awal ke blok teks M, diberikan sebelumnya, kita dapatkan

M = 0000 0001 0010 0011 0100 0101 0110 0111 1000 1001 1010 1011 1100 1101 1110 1111

IP = 1100 1100 0000 0000 1100 1100 1111 1111 1111 0000 1010 1010 1111 0000 1010 1010

Di sini bit ke 58 dari M adalah "1", yang menjadi bit pertama IP. Bit ke-50 dari M adalah "1", yang menjadi bit kedua IP. Bit ke-7 dari M adalah "0", yang menjadi bit terakhir dari IP.

Selanjutnya bagikan IP blok permutasi ke dalam setengah kiri L0 dari 32 bit, dan setengah kanan R0 dari 32 bit.

Contoh: Dari IP, kita mendapatkan L0 dan R0

L0 = 1100 1100 0000 0000 1100 1100 1111 1111

R0 = 1111 0000 1010 1010 1111 0000 1010 1010

Sekarang kita lanjutkan melalui 16 iterasi, untuk 1 <= n <= 16, dengan menggunakan fungsi f yang beroperasi pada dua blok - blok data 32 bit dan kunci Kn dari 48 bit - untuk menghasilkan blok 32 bit. Misalkan + menunjukkan penambahan XOR, (modulo 2 demi bit). Kemudian untuk n pergi dari 1 sampai 16 kita hitung

Ln = Rn-1

Rn = Ln-1 + f (Rn-1, Kn)

Hal ini menghasilkan blok akhir, untuk n = 16, dari L16R16. Artinya, dalam setiap iterasi, kita mengambil 32 bit kanan dari hasil sebelumnya dan membuat mereka menjadi 32 bit kiri dari langkah saat ini. Untuk 32 bit kanan pada langkah saat ini, kita XOR menyisakan 32 bit dari langkah sebelumnya dengan perhitungan f.

Contoh: Untuk n = 1, kita punya

K1 = 000110 110000 001011 101111 111111 000111 000001 110010

L1 = R0 = 1111 0000 1010 1010 1111 0000 1010 1010

R1 = L0 + f (R0, K1)

Masih harus dijelaskan bagaimana fungsi f bekerja. Untuk menghitung f, pertama kita kembangkan setiap blok Rn-1 dari 32 bit menjadi 48 bit. Hal ini dilakukan dengan menggunakan tabel pilihan yang mengulang beberapa bit pada Rn-1. Kita akan memanggil penggunaan tabel pilihan ini fungsi E. Jadi E (Rn-1) memiliki blok input 32 bit, dan blok output 48 bit.

Misalkan E sedemikian rupa sehingga 48 bit outputnya, ditulis sebagai 8 blok masing-masing 6 bit, diperoleh dengan memilih bit inputnya sesuai tabel berikut:

E BIT-SELECTION TABLE

32 1 2 3 4 5

4 5 6 7 8 9

8 9 10 11 12 13

12 13 14 15 16 17

16 17 18 19 20 21

20 21 22 23 24 25

24 25 26 27 28 29

28 29 30 31 32 1

Jadi tiga bit pertama dari E (Rn-1) adalah bit pada posisi 32, 1 dan 2 dari Rn-1 sementara 2 bit terakhir dari E (Rn-1) adalah bit pada posisi 32 dan 1.

Contoh: Kami menghitung E (R0) dari R0 sebagai berikut:

R0 = 1111 0000 1010 1010 1111 0000 1010 1010

E (R0) = 011110 100001 010101 010101 011110 100001 010101 010101

(Perhatikan bahwa setiap blok dari 4 bit asli telah diperluas ke blok 6 bit output.)

Selanjutnya dalam perhitungan f, kita XOR output E (Rn-1) dengan kunci Kn:

Kn + E (Rn-1).

Contoh: Untuk K1, E (R0), kita punya

K1 = 000110 110000 001011 101111 111111 000111 000001 110010

E (R0) = 011110 100001 010101 010101 011110 100001 010101 010101

K1 + E (R0) = 011000 010001 011110 111010 100001 100110 010100 100111.

Kami belum selesai menghitung fungsi f. Sampai titik ini kita telah memperluas Rn-1 dari 32 bit menjadi 48 bit, menggunakan tabel seleksi, dan XORed hasilnya dengan tombol Kn. Kami sekarang memiliki 48 bit, atau delapan kelompok dengan enam bit. Kami sekarang melakukan sesuatu yang aneh dengan masing-masing kelompok terdiri dari enam bit: kami menggunakannya sebagai alamat dalam tabel yang disebut "Kotak S". Setiap kelompok enam bit akan memberi kami alamat dalam kotak S yang berbeda. Terletak di alamat itu akan menjadi nomor 4 bit. Nomor 4 bit ini akan menggantikan 6 bit asli. Hasil akhirnya adalah bahwa delapan kelompok dari 6 bit diubah menjadi

delapan kelompok dari 4 bit (output 4-bit dari kotak S) untuk total 32 bit. Tuliskan hasil sebelumnya, yaitu 48 bit, dalam bentuk:

Kn + E (Rn-1) = B1B2B3B4B5B6B7B8,

Dimana masing-masing Bi adalah kelompok enam bit. Kami sekarang menghitung

S1 (B1) S2 (B2) S3 (B3) S4 (B4) S5 (B5) S6 (B6) S7 (B7) S8 (B8)

dimana Si (Bi) mengacu pada output dari kotak S ke-i.

Untuk mengulang, masing-masing fungsi S1, S2, ..., S8, mengambil blok 6-bit sebagai masukan dan menghasilkan blok 4-bit sebagai keluaran. Tabel untuk menentukan S1 ditunjukkan dan dijelaskan di bawah ini:

S1

Column Number

Row No.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
1	0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8
2	4	1	14	8	13	6	2	11	15	12	9	7	3	10	5	0
3	15	12	8	2	4	9	1	7	5	11	3	14	10	0	6	13

Jika S1 adalah fungsi yang didefinisikan dalam tabel ini dan B adalah blok 6 bit, maka S1 (B) ditentukan sebagai berikut: Bit pertama dan terakhir B mewakili basis 2 angka di kisaran desimal 0 sampai 3 (atau biner 00 sampai 11). Biarkan nomor itu menjadi i. Bagian tengah 4 bit B mewakili di basis 2 angka di kisaran desimal 0 sampai 15 (biner 0000 sampai 1111). Biarkan nomor itu j. Cari di atas meja nomor di baris ke-i dan kolom ke-j. Ini adalah angka di kisaran 0 sampai 15 dan secara unik ditunjukkan oleh blok 4 bit. Blok itu adalah output S1 (B) S1 untuk input B. Sebagai contoh, untuk blok input B = 011011 bit pertama adalah "0" dan bit terakhir "1" memberi 01 sebagai barisnya. Ini adalah baris 1. Empat bit tengah adalah "1101". Ini adalah bilangan biner yang setara dengan desimal 13, jadi kolomnya adalah kolom nomor 13. Pada baris 1, kolom 13 muncul 5. Ini menentukan output; 5 adalah biner 0101, sehingga hasilnya 0101. Maka S1 (011011) = 0101.

Tabel yang menentukan fungsi S1, ..., S8 adalah sebagai berikut:

S1

14	4	13	1	2	15	11	8	3	10	6	12	5	9	0	7
0	15	7	4	14	2	13	1	10	6	12	11	9	5	3	8

4 1 14 8 13 6 2 11 15 12 9 7 3 10 5 0

15 12 8 2 4 9 1 7 5 11 3 14 10 0 6 13

S2

15 1 8 14 6 11 3 4 9 7 2 13 12 0 5 10

3 13 4 7 15 2 8 14 12 0 1 10 6 9 11 5

0 14 7 11 10 4 13 1 5 8 12 6 9 3 2 15

13 8 10 1 3 15 4 2 11 6 7 12 0 5 14 9

S3

10 0 9 14 6 3 15 5 1 13 12 7 11 4 2 8

13 7 0 9 3 4 6 10 2 8 5 14 12 11 15 1

13 6 4 9 8 15 3 0 11 1 2 12 5 10 14 7

1 10 13 0 6 9 8 7 4 15 14 3 11 5 2 12

S4

7 13 14 3 0 6 9 10 1 2 8 5 11 12 4 15

13 8 11 5 6 15 0 3 4 7 2 12 1 10 14 9

10 6 9 0 12 11 7 13 15 1 3 14 5 2 8 4

3 15 0 6 10 1 13 8 9 4 5 11 12 7 2 14

S5

2 12 4 1 7 10 11 6 8 5 3 15 13 0 14 9

14 11 2 12 4 7 13 1 5 0 15 10 3 9 8 6

4 2 1 11 10 13 7 8 15 9 12 5 6 3 0 14

11 8 12 7 1 14 2 13 6 15 0 9 10 4 5 3

S6

12 1 10 15 9 2 6 8 0 13 3 4 14 7 5 11

10 15 4 2 7 12 9 5 6 1 13 14 0 11 3 8

9 14 15 5 2 8 12 3 7 0 4 10 1 13 11 6

4 3 2 12 9 5 15 10 11 14 1 7 6 0 8 13

S7

4 11 2 14 15 0 8 13 3 12 9 7 5 10 6 1

13 0 11 7 4 9 1 10 14 3 5 12 2 15 8 6

1 4 11 13 12 3 7 14 10 15 6 8 0 5 9 2

6 11 13 8 1 4 10 7 9 5 0 15 14 2 3 12

S8

13 2 8 4 6 15 11 1 10 9 3 14 5 0 12 7

1 15 13 8 10 3 7 4 12 5 6 11 0 14 9 2

7 11 4 1 9 12 14 2 0 6 10 13 15 3 5 8

2 1 14 7 4 10 8 13 15 12 9 0 3 5 6 11

Contoh: Untuk putaran pertama, kita dapatkan sebagai output dari delapan kotak S:

K1 + E (R0) = 011000 010001 011110 111010 100001 100110 010100 100111.

S1 (B1) S2 (B2) S3 (B3) S4 (B4) S5 (B5) S6 (B6) S7 (B7) S8 (B8) = 0101 1100 1000 0010 1011 0101 1001 0111

Tahap akhir dalam perhitungan f adalah dengan melakukan permutasi P dari keluaran S-box untuk mendapatkan nilai akhir dari f:

f = P (S1 (B1) S2 (B2) ... S8 (B8))

Permutasi P didefinisikan dalam tabel berikut. P menghasilkan output 32-bit dari input 32-bit dengan menggerakkan bit blok input.

P

16 7 20 21

29 12 28 17

1	15	23	26
5	18	31	10
2	8	24	14
32	27	3	9
19	13	30	6
22	11	4	25

Contoh: Dari output dari delapan kotak S:

S1 (B1) S2 (B2) S3 (B3) S4 (B4) S5 (B5) S6 (B6) S7 (B7) S8 (B8) = 0101 1100 1000 0010 1011 0101 1001 0111

kita mendapatkan

f = 0010 0011 0100 1010 1010 1001 1011 1011

R1 = L0 + f (R0, K1)

= 1100 1100 0000 0000 1100 1100 1111 1111

+ 0010 0011 0100 1010 1010 1001 1011 1011

= 1110 1111 0100 1010 0110 0101 0100 0100

Pada putaran berikutnya, kita akan memiliki L2 = R1, yang merupakan blok yang baru kita hitung, dan kemudian kita harus menghitung R2 = L1 + f (R1, K2), dan seterusnya untuk 16 putaran. Pada akhir ronde keenam belas kita memiliki blok L16 dan R16. Kami kemudian membalik urutan dua blok ke blok 64-bit

R16L16

dan menerapkan IP^{-1} permutasi akhir seperti yang didefinisikan oleh tabel berikut:

IP^{-1}

40	8	48	16	56	24	64	32
39	7	47	15	55	23	63	31
38	6	46	14	54	22	62	30
37	5	45	13	53	21	61	29
36	4	44	12	52	20	60	28
35	3	43	11	51	19	59	27
34	2	42	10	50	18	58	26

33 1 41 9 49 17 57 25

Artinya, output dari algoritma tersebut memiliki bit 40 dari blok preoutput sebagai bit pertama, bit 8 sebagai bit kedua, dan seterusnya, sampai bit 25 dari blok preoutput adalah bit terakhir dari output.

Contoh: Jika kita memproses 16 blok menggunakan metode yang didefinisikan sebelumnya, kita dapatkan, pada ronde ke-16,

L16 = 0100 0011 0100 0010 0011 0010 0011 0100

R16 = 0000 1010 0100 1100 1101 1001 1001 0101

membalik urutan dua blok ini dan menerapkan permutasi terakhir

R16L16 = 00001010 01001100 11011001 10010101 01000011 01000010 00110010 00110100

IP^{-1} = 10000101 11101000 00010011 01010100 00001111 00001010 10110100 00000101

yang dalam format heksadesimal ini

85E813540F0AB405.

Ini adalah bentuk terenkripsi M = 0123456789ABCDEF: yaitu, C = 85E813540F0AB405.

Dekripsi hanyalah kebalikan dari enkripsi, mengikuti langkah yang sama seperti di atas, namun membalik urutan di mana subkunci diterapkan.

c. Kelebihan & Kekurangan DES

Data Encryption Standard (DES) pun seperti sistem yang lainnya, memiliki kekurangan dan kelebihan. Salah satu kekurangan DES adalah proses yang lebih lama dalam melakukan proses dekripsi dan enkripsi.

Kelebihan DES yaitu, sistem sandi lebih kompleks sehingga membutuhkan waktu yang tidak cepat untuk menembus enkripsi DES dan sulit untuk diketahui dari pihak luar. Tetapi setelah berkembangannya jaman DES tidak digunakan karena ukuran kunci yang terlalu kecil, sehingga mudah untuk ditembus

2. Triple DES

a. Fungsi Triple DES

Pada awalnya, ukuran kunci sandi DES yaitu 56 bit sudah mencukupi pada saat algoritme ini dibuat. Namun, dengan meningkatnya kemampuan komputasi, serangan brutal telah mungkin terjadi. Triple DES menyediakan metode yang sederhana dengan menambah ukuran kunci DES untuk mencegah serangan tersebut, tanpa memerlukan perancangan sandi blok (block cipher) yang sama sekali baru.

Triple-DES hanya DES dengan dua kunci 56-bit yang diaplikasikan. Dengan pesan plaintext, kunci pertama digunakan untuk mengenkripsi pesan. Kunci kedua digunakan untuk DES-mendekripsi pesan terenkripsi. (Karena kunci kedua bukanlah kunci yang benar, dekripsi ini hanya mengacak data lebih jauh.) Pesan dua kali tergesa-gesa kemudian dienkripsi lagi dengan kunci pertama untuk menghasilkan cipherteks terakhir. Prosedur tiga langkah ini disebut triple-DES.

Triple-DES hanya DES dilakukan tiga kali dengan dua kunci yang digunakan dalam urutan tertentu. (Triple-DES juga bisa dilakukan dengan tiga kunci terpisah, bukan hanya dua. Dalam kedua kasus, ruang kunci yang dihasilkan adalah sekitar 2 ^ 112.).

b. Cara Kerja Triple DES

Tahap pertama, plainteks yang diinputkan dioperasikan dengan kunci eksternal pertama (K1) dan melakukan proses enkripsi dengan menggunakan algoritma DES. Sehingga menghasilkan pra-cipherteks pertama. Tahap kedua, pra-cipherteks pertama yang dihasilkan pada tahap pertama, kemudian dioperasikan dengan kunci eksternal kedua (K2) dan melakukan proses enkripsi atau proses dekripsi (tergantung cara pengenkripsian yang digunakan) dengan menggunakan algoritma DES. Sehingga menghasilkan prs-cipherteks kedua. Tahap terakhir, pra-cipherteks kedua yang dihasilkan pada tahap kedua, dioperasikan dengan kunci eksternal ketiga (K3) dan melakukan proses enkripsi dengan menggunakan algoritma DES, sehingga menghasilkan cipherteks (C).

Dalam kriptografi, Triple DES adalah nama umum untuk Algoritma Data Encryption Triple (TDEA atau Triple DEA) blok cipher, yang menerapkan Standar Enkripsi Data (DES) algoritma cipher tiga kali untuk setiap blok data.

Ukuran kunci DES asli cipher dari 56 bit pada umumnya cukup ketika algoritma yang dirancang, tetapi ketersediaan daya komputasi semakin membuat serangan brute force layak. Triple DES menyediakan metode yang relatif sederhana meningkatkan ukuran kunci DES untuk melindungi terhadap serangan tersebut, tanpa perlu merancang sebuah algoritma blok cipher baru.

Seperti semua blok cipher, enkripsi dan dekripsi dari beberapa blok data dapat dilakukan dengan menggunakan berbagai modus operasi, yang secara umum dapat didefinisikan secara independen dari algoritma blok cipher. Namun, ANS X9.52 menentukan secara langsung, dan NIST SP 800-67 menentukan melalui SP 800-38A bahwa beberapa mode hanya dapat digunakan dengan batasan tertentu pada mereka yang belum tentu berlaku untuk spesifikasi umum dari mode. Sebagai contoh, ANS X9.52 menetapkan bahwa untuk chaining blok cipher, vektor inisialisasi akan berbeda setiap kali, sedangkan ISO / IEC 10116 tidak. PUB FIPS 46-3 dan ISO / IEC 18033-3 mendefinisikan hanya algoritma blok tunggal, dan tidak menempatkan batasan pada mode operasi untuk beberapa blok

c. Kelebihan & Kekurangan Triple DES

Triple DES memiliki kekurangan pada performansi yang lambat dalam software.

Kelebihan yang dimiliki oleh Triple DES adalah membuat menjebol enkripsinya sangat lama

3. AES & RSA

a. Fungsi AES & RSA

Advanced Encryption Standard (AES) merupakan algoritma cryptographic yang dapat digunkan untuk mengamakan data. Algoritma AES adalah blokchipertext simetrik yang dapat mengenkripsi (encipher) dan dekripsi (decipher) infoermasi. Enkripsi merubah data yang tidak dapat lagi dibaca disebut ciphertext; sebaliknya dekripsi adalah merubah ciphertext data menjadi bentuk semula yang kita kenal sebagai plaintext. Algoritma AES is mengunkan kunci kriptografi 128, 192, dan 256 bits untuk mengenkrip dan dekrip data pada blok 128 bits. Sedangkan Algoritma RSA masih digunakan secara luas dalam protocol electronic commerce dan dipercaya dalam pengamanan dengan kunci yang

sangat panjang. Algoritma RSA disebut sebagai kunci publik karena kunci enkripsi dapat dibuat public yang berarti semua orang dapat mengetahuinya. Walaupun dibuat public key, keamanan algoritma RSA sangat terjaga. Hal itu dikarenakan kunci yang digunakan untuk enkripsi pada algoritma RSA berbeda dengan kunci yang digunakan untuk dekripsinya. Keamanan enkripsi dan dekripsi algoritma RSA terletak pada kesulitan untuk memfaktorkan modulus n yang sangat besar. Penamaan algoritma RSA diambil dari nama penemunya, yaitu Rivest, Shamir dan Adleman yang diplubikasikan pada tahun 1977 di MIT yang bertujuan untuk menjawab tantangan dari Algoritma Pertukaran Kunci Diffie Helman.

b. Cara Kerja AES & RSA

Berikut ini adalah operasi Rijndael (AES) yang menggunakan 128 bit kuncii

- Ekspansi kunci utama (dari 128 bit menjadi 1408 bit).
- Pencampuran subkey.
- Ulang dari i=1 sampai i=10 Transformasi : ByteSub (subtitusi per byte) ShiftRow (Pergeseren byte perbaris) MixColumn (Operasi perkalian GF(2) per kolom) Pencampuran subkey (dengan XOR).
- Transformasi : ByteSub dan ShiftRow
- Pencampuran subkey Kesimpulan yang didapat adalah :
    - ✓ AES terbukti kebal menghadapi serangan konvensional (linear dan diferensial attack) yang menggunakan statistik untuk memecahkan sandi.
    - ✓ Kesederhanaan AES memberikan keuntungan berupa kepercayaan bahwa AES tidak ditanami trapdoor.
    - ✓ Namun, kesederhanaan struktur AES juga membuka kesempatan untuk mendapatkan persamaan aljabar AES yang selanjutnya akan diteliti apakah persamaan tersebut dapat dipecahkan
    - ✓ Bila persamaan AES dapat dipecahkan dengan sedikit pasangan plaintext/ciphertext, maka riwayat AES akan berakhir.
    - ✓ AES didesain dengan sangat hati-hati dan baik sehingga setiap komponennya memiliki tugas yang jelas
    - ✓ AES memiliki sifat cipher yang diharapkan yaitu : tahan menghadapi analisis sandi yang diketahui, fleksibel digunakan dalam berbagai perangkat keras dan lunak, baik digunakan untuk fungsi hash karena tidak memiliki weak(semi weak) key, cocok untuk perangkat yang membutuhkan key agility yang cepat, dan cocok untuk stream cipher.

Sedangkan untuk cara kerja RSA. RSA mempunyai dua kunci, yaitu kunci publik dan kunci rahasia. Kunci publik boleh diketahui oleh siapa saja, dan digunakan untuk proses enkripsi. Sedangkan kunci rahasia hanya pihak-pihak tertentu saja yang boleh mengetahuinya, dan digunakan untuk proses dekripsi. Keamanan sandi RSA terletak pada sulitnya memfaktorkan bilangan yang besar. Sampai saat ini RSA masih dipercaya dan digunakan secara luas di internet.

Algoritma Kunci Publik Sandi RSA terdiri dari tiga proses, yaitu proses pembentukan kunci, proses enkripsi dan proses deskripsi. Sebelumnya diberikan terlebih dahulu beberapa konsep perhitungan matematis yang digunakan RSA.

c. Kelebihan & Kekurangan AES & RSA

Kelebihan AES menggunakan struktur SPN (Substitution Permutation Network) yang memiliki derajat paralelisme yang lebih besar, sehingga diharapkan lebih cepat dari pada Feistel.

Kekurangan AES umumnya (termasuk pada Rijndael) adalah berbedanya struktur enkripsi dan dekripsi sehingga diperlukan dua algoritma yang berbeda untuk enkripsi dan dekripsi. Dan tentu pula tingkat keamanan enkripsi dan dekripsinya menjadi berbeda. AES memiliki blok masukan dan keluaran serta kunci 128 bit.

Kelebihan & Kekurangan Algoritma RSA

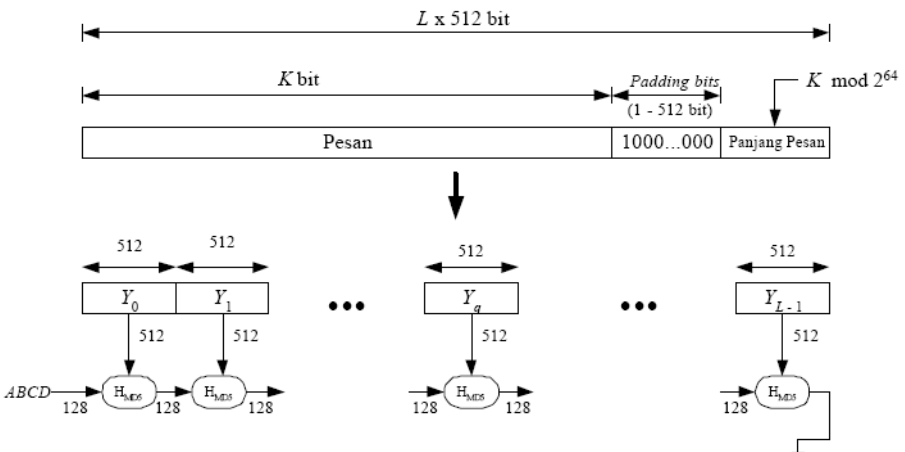
Kekuatan algoritma RSA terletak pada tingkat kesulitan dalam memfaktorkan bilangan menjadi faktor primanya, dalam hal ini memfaktorkan n menjadi p dan q. Karena sekali n berhasil difaktorkan, maka menghitung nilai m adalah perkara mudah. Selanjutnya, walau nilai e diumumkan, perhitungan kunci d tidaklah mudah pula karena nilai m yang tidak diketahui. Kelebihan lain algoritma RSA terletak pada ketahanannya terhadap berbagai bentuk serangan, terutama serangan brute force. Hal ini dikarenakan kompleksitas dekripsinya yang dapat ditentukan secara dinamis dengan cara menentukan nilai p dan q yang besar pada saat proses pembangitkan pasangan kunci, sehingga dihasilakan sebuah key space yang cukup besar, sehingga tahan terhadap serangan.

Namun demikian, kelebihan tersebut sekaligus menjadi kelemahan dari sistem ini. Ukuran kunci privat yang terlalu besar akan mengakibatkan proses dekripsi yang cukup lambat, terutama untuk ukuran pesan yang besar. Oleh karena itu, RSA umumnya digunakan untuk meng-enkripsi pesan berukuran kecil seperti kata kunci dari enkripsi simetris seperti DES dan AES yang kemudian kunci tersebut dikirim secara bersamaan dengan pesan utama.

4. MD5, SHA1
   a. Fungsi MD5, SHA1
      Algoritma yang menggunakan fungsi hash satu arah yang diciptakan oleh Ron Rivest. Algoritma merupakan pengembangan dari algoritma-algoritma sebelumnya yaitu algoritma MD2 dan algoritma MD4 karena kedua algoritma ini berhasil diserang para cryptanalist.
   b. Cara Kerja MD5, SHA1

      Cara kerja kriptografi algoritma MD5

      Cara kerja kriptografi algoritma MD5 adalah menerima input berupa pesan dengan ukuran sembarang dan menghasilkan message diggest yang memiliki panjang 128 bit. Berikut ilustrasi gambar dari pembuatan message diggest pada kriptografi algoritma MD5 :



Menilik dari gambar diatas, secara garis besar pembuatan message digest ditempuh melalui empat langkah, yaitu :

1.Penambahan bit bit pengganjal

Proses pertama yang dilakukan adalah menambahkan pesan dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 modulo 512. Ini berarti setelah menambahkan bit-bit pengganjal, kini panjang pesan adalah 64 bit kurang dari kelipatan 512. Hal yang perlu diingat adalah angka 512 muncul karena algoritma MD5 memproses pesan dalam blok-blok yang berukuran 512.

Apabila terdapat pesan dengan panjang 448 bit, maka pesan tersebut akan tetap ditambahkan dengan bit-bit pengganjal. Pesan akan ditambahkan dengan 512 bit menjadi 96 bit. Jadi panjang bit-bit pengganjal adalah antara 1 sampai 512. Lalu satu hal lagi yang perlu diperhatikan adalah bahwasanya bit-bit pengganjal terdiri dari sebuah bit 1 diikuti dengan sisanya bit 0.

2.Penambahan nilai panjang pesan semula

kemudian proses berikutnya adalah pesan ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Apabila panjang pesan lebih besar dari 264 maka yang diambil adalah panjangnya dalam modulo 264. dengan kata lain, jika pada awalnya panjang pesan sama dengan K bit, maka 64 bit yang ditambahkan menyatakan K modulo 264. sehingga setelah proses kedua ini selesai dilakukan maka panjang pesan sekarang adalah 512 bit.

3.Inisialisasi penyangga MD5

Pada algoritma MD5 dibutuhkan empat buah penyangga atau buffer, secara berurut keempat nama penyangga diberi nama A, B, C dan D. Masing-masing penyangga memiliki panjang 32 bit. Sehingga panjang total :

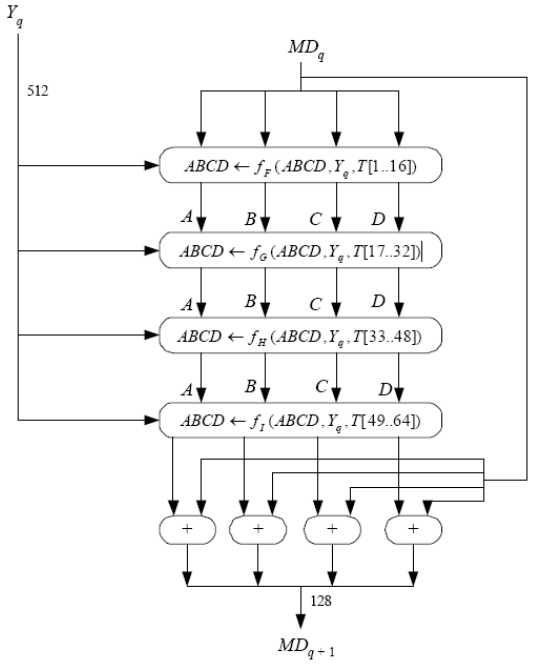
A =	32 bit
B =	32 bit
C =	32 bit
D =	32 bit +
total =	128 bit

Keempat penyangga ini menampung hasil antara dan hasil akhir. Setiap penyangga diinisialisasi dengan nilai-nilai (dalam notasi Hexadesimal) sebagai berikut :

A = 01234567
B = 89ABCDEF
C = FEDCBA98
D = 76543210

4.Pengolahan pesan dalam blok berukuran 512 bit

Proses berikutnya adalah pesan dibagi menjadi L buah blok yang masing-masing panjangnya 512 bit (Y_0 sampai Y_{L-1}). Setelah itu setiap blok 512 bit diproses bersama dengan penyangga MD yang menghasilkan keluaran 128 bit, dan ini disebut H_{MD5}. Berikut ini gambaran dari proses H_{MD5} :



Dari gambar diatas dapat kita lihat bahwa proses H_{MD5} terdiri dari 4 buah putaran, dan masing-masing putaran melakukan opersi dasar MD5 sebanyak 16 kali. Dimana disetiap operasi dasar memakai sebuah elemen T. Sehingga setiap putaran memakai 16 elemen tabel T.

Cara Kerja SH1

Pesan diberi tambahan untuk membuat panjangnya menjadi kelipatan 512 bit (l x 512). Jumlah bit asal adalah k bit. Tambahkan bit secukupnya sampai 64 bit kurangnya dari kelipatan 512 (512 – 64 = 448), yang disebut juga kongruen dengan 448 (mod 512). Kemudian tambahkan 64 bit yang menyatakan panjang pesan. Inisiasi 5 md variable dengan panjang 32 bit yaitu a,b,c,d,e. Pesan dibagi menjadi blok-blok berukuran 512 bit dan setiap blok diolah. Kemudian keluaran setiap blok digabungkan dengan keluaran blok berikutnya, sehingga diperoleh output (diggest). Fungsi kompresi yang digunakan oleh algoritma sha-1 adalah sebagai berikut : A,b,c,d,e ← (e + f (t,b,c,d) + s5 (a) + wt + kt),a,s30(b),c,d.]

3. Kelebihan & Kekurangan MD5, SHA1

1. Keamanan terhadapserangan brute-force. Hal yang paling penting adalah bahwa SHA-1 menghasilkan diggest 32-bit lebih panjang dari MD5. Dengan brute-force maka SHA-1 lebihkuat dibanding MD5.

2. Keamanan terhadap kriptanalisis. Kelemahan MD5 ada pada design sehingga lebih mudah dilakukan kriptanalisis dibandingkan SHA-1

3. Kecepatan. Kedua algoritma bekerja pada modulo 232 sehingga keduanya bekerja baik pada arsitektur 32 bit. SHA-1 mempunyai langkah lebih banyak dibandingkan MD5 (80 dibanding MD5 64) dan harus memproses 160 bit buffer dibanding DM5 128 bit buffer, sehingga SHA-1 bekerja lebih lambat dibanding MD5 pada perangkat keras yang sama.

4.Simplicity. Kedua algoritma simple untuk dijelaskan dan mudah untuk diiemplementasikan karena tidak membutuhkan program yang besar atau table subtitusi yang besar pula.

5. Digital Signature

a. Fungsi Digital Signature

Sistem ini berfungsi untuk memberikan jaminan bahwa dokumen digital adalah otentik. Menambahkan sistem Digital signature proses transfer dokumen akan terjamin otentikasi dan kerahasiannya. Dengan demikian dapat menghindari kemungkinan terjadinya pemalsuan dokumen terkait.

b. Cara Kerja Digital Signature

Memanfaatkan dua buah kunci, yaitu kunci publik dan kunci privat. Kunci publik digunakan untuk mengenkripsi data, sedangkan kunci privat digunakan untuk mendekripsi data. Pertama, dokumen di-hash dan menghasilkan Message Digest. Kemudian, Message Digest dienkripsi oleh kunci publik menjadi Digital Signature. Untuk membuka Digital Signature tersebut diperlukan kunci privat. Bila data telah diubah oleh pihak luar, maka Digital Signature juga ikut berubah sehingga kunci privat yang ada tidak akan bisa membukanya. Ini merupakan salah satu syarat keaman jaringan, yaitu Authenticity. Artinya adalah, keaslian data dapat terjamin dari perubahan-perubahan yang dilakukan pihak luar. Dengan cara yang sama, pengirim data tidak dapat menyangkal data yang telah dikirimkannya. Bila Digital Signature cocok dengan kunci privat yang dipegang oleh penerima data, maka dapat dipastikan bahwa pengirim adalah pemegang kunci privat yang sama. Ini berarti Digital Signature memenuhi salah satu syarat keamanan jaringan, yaitu Nonrepudiation atau non-penyangkalan.

c. Kelebihan & Kekurangan Digital Signature

Kelebihan :

- Algoritma ini dirancang sehingga proses enkripsi/dekripsi membutuhkan waktu yang singkat.
- Ukuran kunci relatif lebih pendek.
- Algoritmanya bisa menghasilkan cipher(sebuah algoritma untuk menampilkan enkripsi dan kebalikannya dekripsi, serangkaian langkah yang terdefinisi yang diikuti sebagai prosedur) yang lebih kuat.
- Autentikasi pengiriman pesan langsung diketahui dari ciphertext yang diterima, karena kunci hanya diketahui oleh pengirim dan penerima pesan saja. Kelemahan :
- Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini.
- Kunci harus sering diubah, mungkin pada setiap sesi komunikasi.

Kekurangan :
- Kunci harus dikirim melalui saluran yang aman. Kedua entitas yang berkomunikasi harus menjaga kerahasiaan kunci ini.
- Kunci harus sering diubah, mungkin pada setiap sesi komunikasi.

Dikumentasi Codingan Program DES, Triple DES, AES & RSA, MD5, SHA1, dan Digital Signature :

- DES
  https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/des.js
- Triple DES
  https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/triple-des.js
- AES & RSA
  AES :
  https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/aes.js
- MD5, SHA1
  https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/sha1.js
  https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/md5.js
  https://github.com/NaufalHSyahputra/CryptoTools/blob/master/proses/sha1-core.js
- Digital Signature
  https://github.com/NaufalHSyahputra/CryptoTools/tree/master/proses/ds